Incremental Data Flow analysis using PRISM

Rashmi Rekha Mech
(*Project Guide: Prof. Uday Khedker*)

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

June 2015

Outline of the talk

- Incremental Data Flow analysis
  - Bit-vector frameworks
  - General frameworks
  - Method to reduce the size of affected region
- Overview of PRISM
- Incremental driver
  - Testing
- Limitations of old Solver
- Future work

Part I

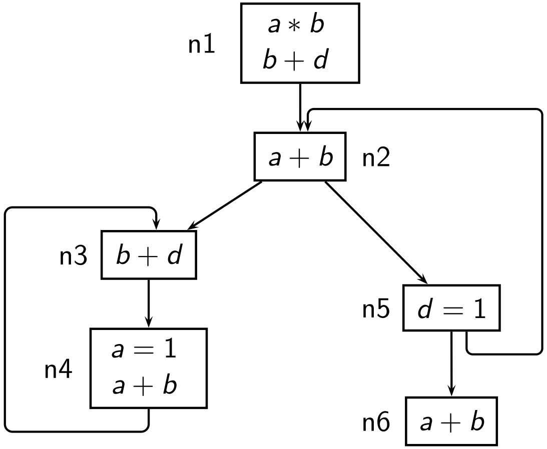
Incremental Data Flow Analysis

Why Incremental Analysis?

When program undergoes changes:

- Some or all computed data flow information become invalid
- Re-computation is required

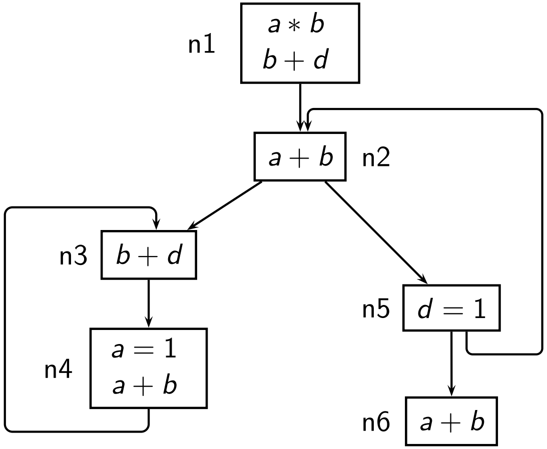
Motivating Example - Available Expression Analysis



Bit Vector $\boxed{a * b \mid b + d \mid a + b}$

Motivating Example - Available Expression Analysis

1st Iteration

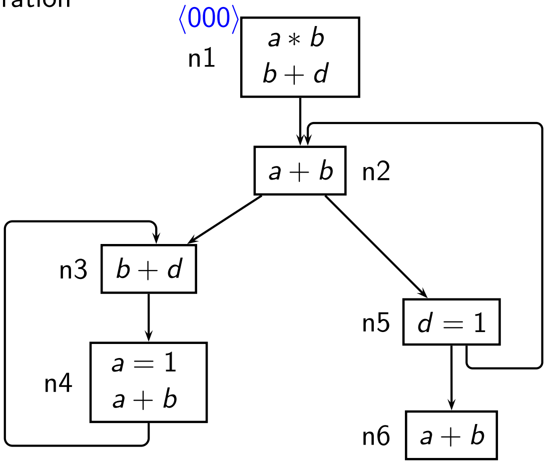


Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

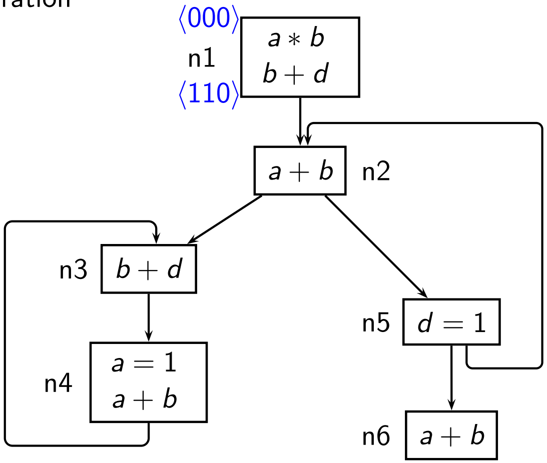
1st Iteration



Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

1st Iteration

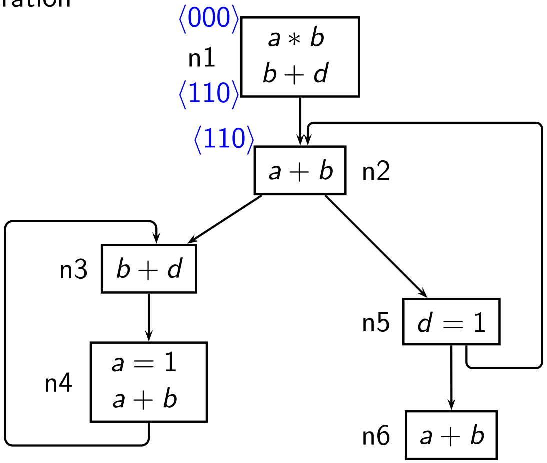


Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

1st Iteration

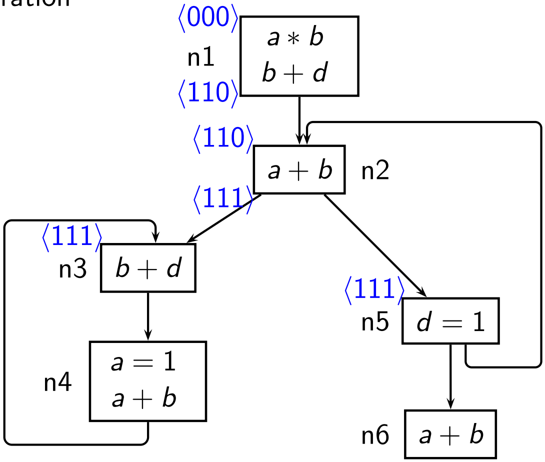


Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

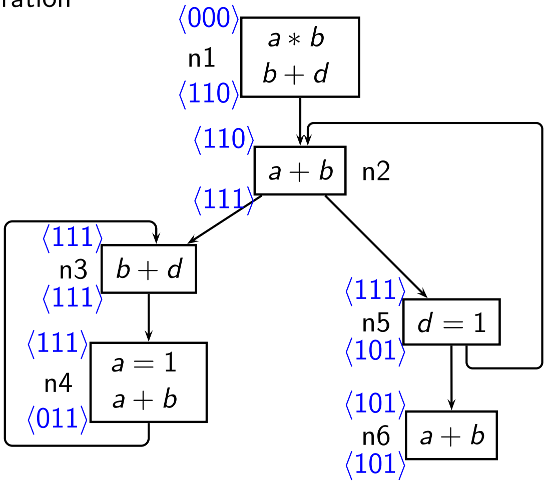
1st Iteration



Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

1st Iteration

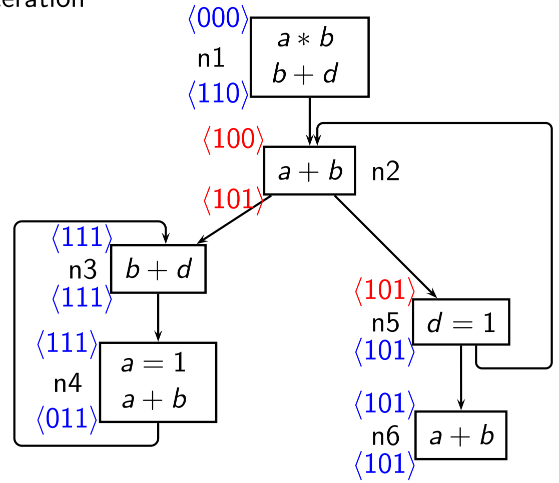


Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

2nd Iteration

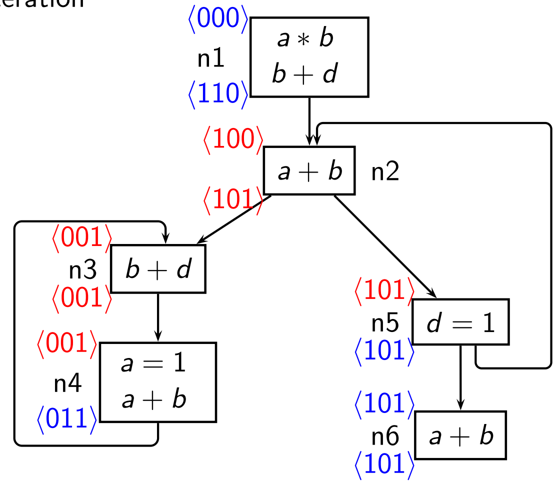


Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

2nd Iteration



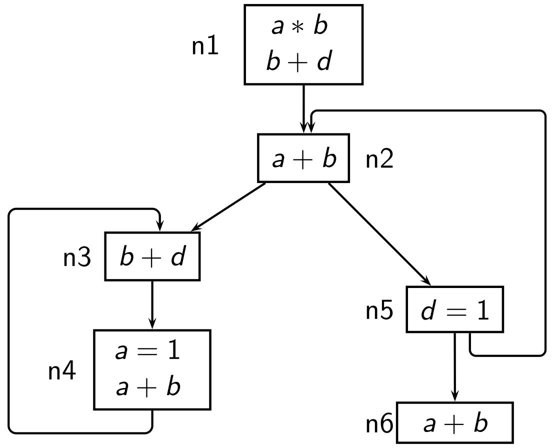
Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

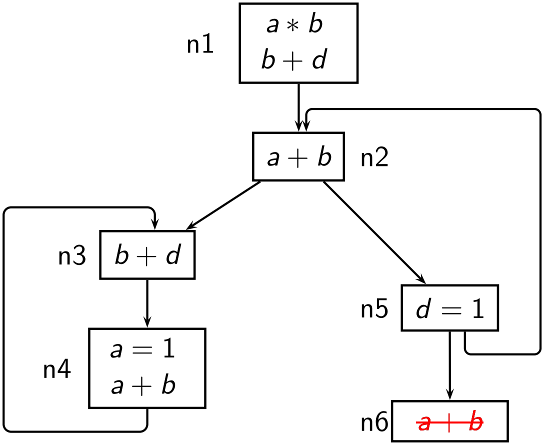
- It requires 3 iterations to converge

Motivating Example - Available Expression Analysis



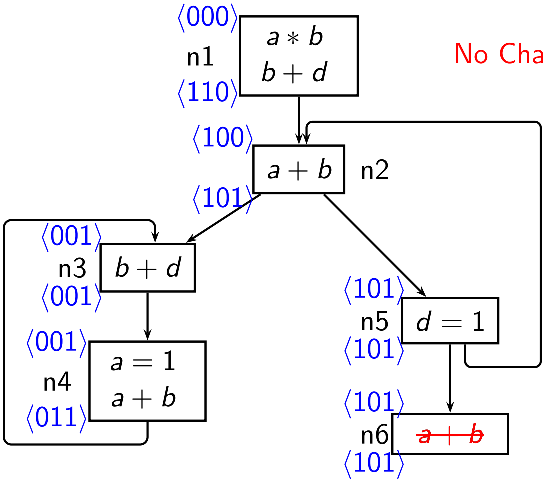
Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis



Bit Vector	$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis



No Change in data flow values

Bit Vector

$a * b$	$b + d$	$a + b$

Motivating Example - Available Expression Analysis

- Recomputing the values from the scratch is very inefficient
- Need an incremental analysis:
  - modifies only affected data flow information
  - more cost effective then **exhaustive** analysis

Part II

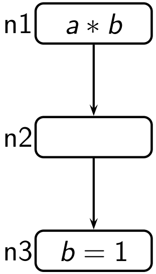
Incremental Analysis in Bit-vector Framework

Flow functions in bit-vector frameworks

- Possible flow functions:
    - Raise : Result is always top
    - Lower : Result is always bottom
    - Propagate : Propagates the value from one program point to another

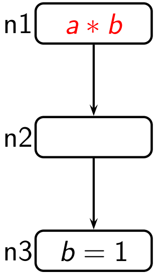
Example for Flow Functions

Available Expression Analysis



Example for Flow Functions

Available Expression Analysis

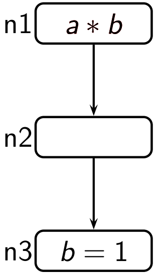


Raise Function

$\text{Gen}_1 = 1$
$\text{Kill}_1 = 0$
$\text{IN}_1 = 0$
$\text{OUT}_1 = \text{Gen}_1 \cup (\text{IN}_1\text{-Kill}_1) = 1$

Example for Flow Functions

Available Expression Analysis



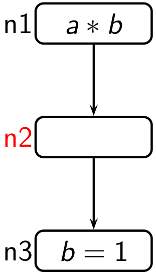
Raise Function

$\text{Gen}_1 = 1$
$\text{Kill}_1 = 0$
$\text{IN}_1 = 0$
$\text{OUT}_1 = \text{Gen}_1 \cup (\text{IN}_1\text{-}\text{Kill}_1) = 1$

Result is always top

Example for Flow Functions

Available Expression Analysis

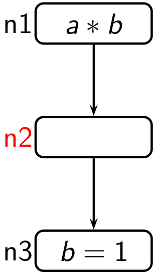


Propagate Function

$\text{Gen}_2 = 0$
$\text{Kill}_2 = 0$
$\text{IN}_2 = 1$
$\text{OUT}_2 = \text{Gen}_2 \cup (\text{IN}_2\text{-Kill}_2) = \text{IN}_2 = 1$

Example for Flow Functions

Available Expression Analysis



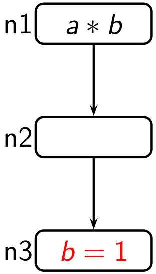
Propagate Function

$\text{Gen}_2 = 0$
$\text{Kill}_2 = 0$
$\text{IN}_2 = 1$
$\text{OUT}_2 = \text{Gen}_2 \cup (\text{IN}_2 - \text{Kill}_2) = \text{IN}_2 = 1$

Propagates the value at IN to OUT

Example for Flow Functions

Available Expression Analysis

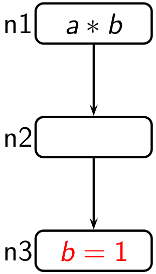

$$\begin{tikzcd}
\text{n1}\ \boxed{a * b} \arrow[d] \\
\text{n2}\ \boxed{} \arrow[d] \\
\text{n3}\ \boxed{b = 1}
\end{tikzcd}$$

Lower Function

$\text{Gen}_3 = 0$
$\text{Kill}_3 = 1$
$\text{IN}_3 = 1$
$\text{OUT}_3 = \text{Gen}_3 \cup (\text{IN}_3\text{-Kill}_3) = 0$

Example for Flow Functions

Available Expression Analysis


$$\begin{tikzcd}
\text{n1}\ \boxed{a * b} \arrow[d] \\
\text{n2}\ \boxed{} \arrow[d] \\
\text{n3}\ \boxed{b = 1}
\end{tikzcd}$$

Lower Function

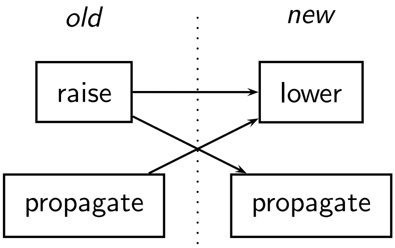
$\text{Gen}_3 = 0$
$\text{Kill}_3 = 1$
$\text{IN}_3 = 1$
$\text{OUT}_3 = \text{Gen}_3 \cup (\text{IN}_3\text{-Kill}_3) = 0$

Result is always bottom

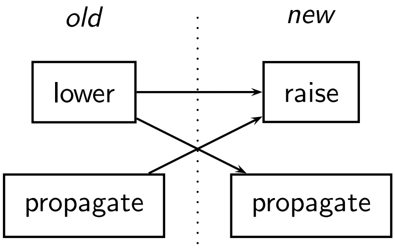
Changes in Bit-vector Frameworks

- As a consequence of some change in a node, some data flow values may:
  - change from top to bottom
  - change from bottom to top
  - remain same

Possible Changes in Flow Functions for Top to Bottom Change



Possible Changes in Flow Functions for Top to Bottom Change

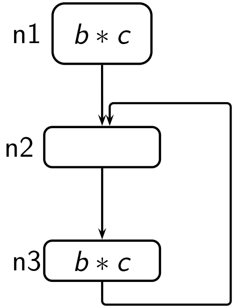


Handling Top to Bottom Change

- Top value is an intermediate value until data flow analysis is completed
- Whenever there is top to bottom change, the changes can be propagated directly to its neighbouring nodes

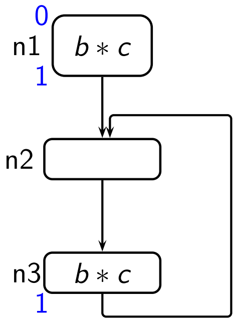
Example for Top to Bottom Change

Initial Available Expression Analysis



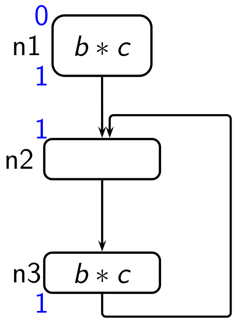
Example for Top to Bottom Change

Initial Available Expression Analysis



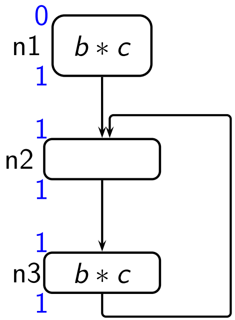
Example for Top to Bottom Change

Initial Available Expression Analysis

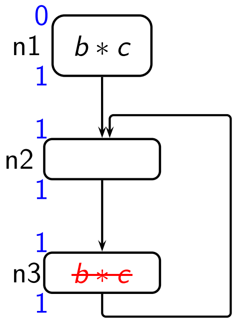


Example for Top to Bottom Change

Initial Available Expression Analysis

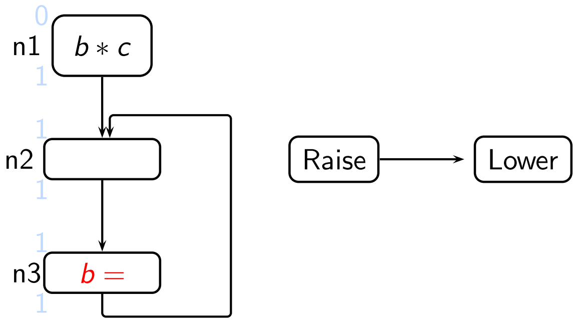


Example for Top to Bottom Change



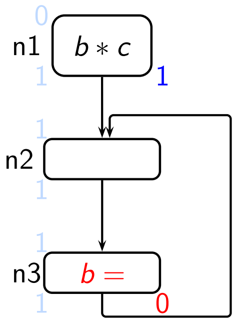
Example for Top to Bottom Change

Top to Bottom change

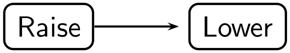


Example for Top to Bottom Change

Top to Bottom change



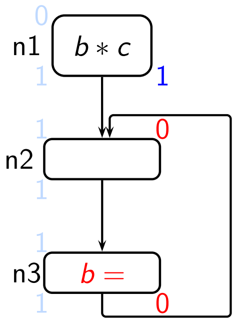
$IN_2 = OUT_1 \cap OUT_3$



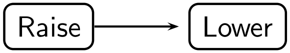
Directly Propagate the change to its neighbour

Example for Top to Bottom Change

Top to Bottom change



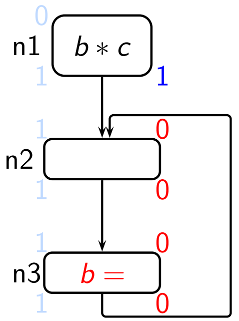
$IN_2 = OUT_1 \cap OUT_3$



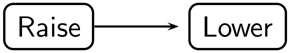
Directly Propagate the change to its neighbour

Example for Top to Bottom Change

Top to Bottom change



$IN_2 = OUT_1 \cap OUT_3$



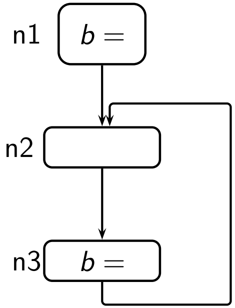
Directly Propagate the change to its neighbour

Handling Bottom to Top Change

- Bottom value is a final value even during analysis
- Whenever there is bottom to top change, we cannot directly propagate the changes to its neighbouring nodes
- Need some more processing

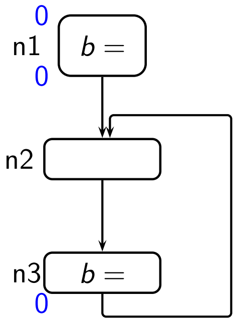
Example for Bottom to Top Change

Initial Available Expression Analysis



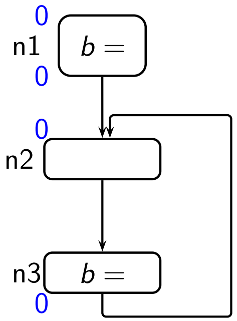
Example for Bottom to Top Change

Initial Available Expression Analysis



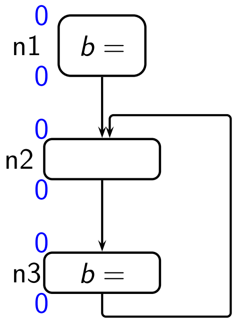
Example for Bottom to Top Change

Initial Available Expression Analysis

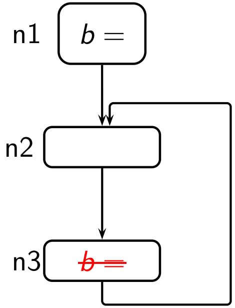


Example for Bottom to Top Change

Initial Available Expression Analysis

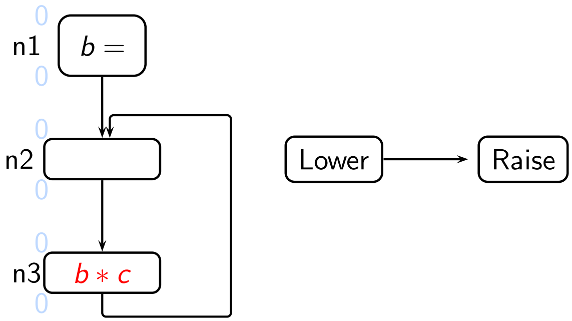


Example for Bottom to Top Change



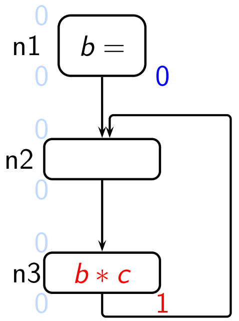
Example for Bottom to Top Change

Bottom to Top change

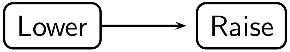


Example for Bottom to Top Change

Bottom to Top change



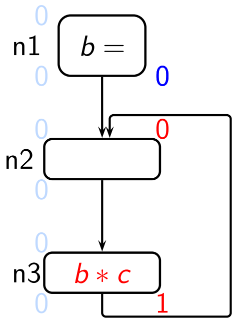
$IN_2 = OUT_1 \cap OUT_3$



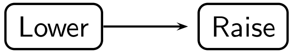
Cannot propagate the change to its neighbouring nodes

Example for Bottom to Top Change

Bottom to Top change



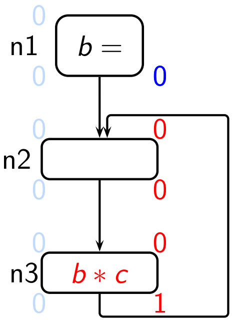
$IN_2 = OUT_1 \cap OUT_3$



Cannot propagate the change to its neighbouring nodes

Example for Bottom to Top Change

Bottom to Top change



$IN_2 = OUT_1 \cap OUT_3$

$$\begin{tikzcd} \boxed{\text{Lower}} \arrow[r] & \boxed{\text{Raise}} \end{tikzcd}$$

Cannot propagate the change to its neighbouring nodes

Bottom to Top Change

- Need some more processing

Bottom to Top Change

- Steps to incorporate bottom to top change:

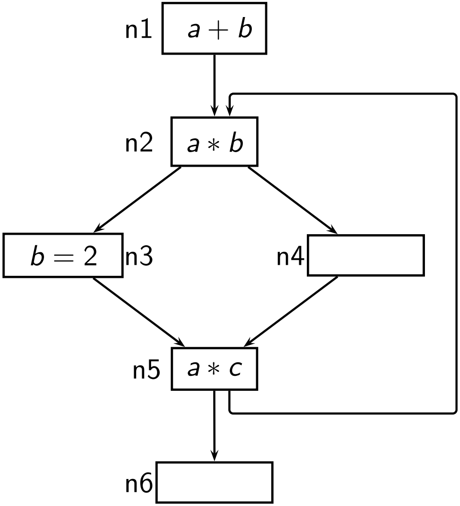
Bottom to Top Change

- Steps to incorporate bottom to top change:
  - Identify the data flow values which may become top

Bottom to Top Change

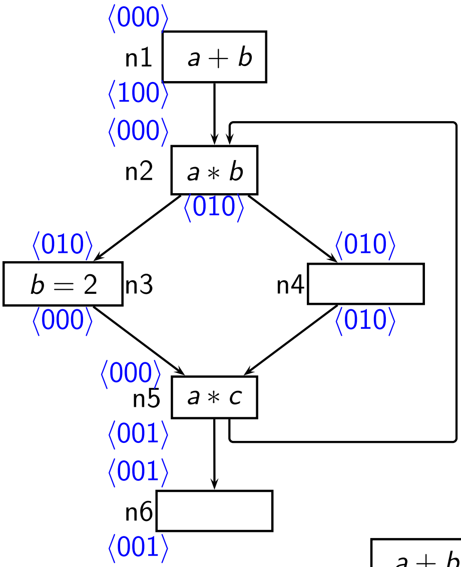
- Steps to incorporate bottom to top change:
  - Identify the data flow values which may become top
  - Find out the data flow values which must remain bottom due to the effect of some other property

Motivating Example



$a + b$	$a * b$	$a * c$

Motivating Example

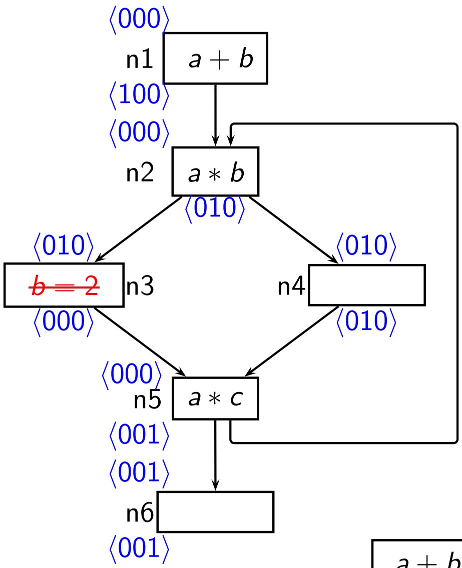


Initial Available Expression Analysis

	$a + b$		$a * b$		$a * c$	
Node	In	Out	In	Out	In	Out
1.	0	1	0	0	0	0
2.	0	0	0	1	0	0
3.	0	0	1	0	0	0
4.	0	0	1	1	0	0
5.	0	0	0	0	0	1
6.	0	0	0	0	1	1

$a + b$	$a * b$	$a * c$

Motivating Example

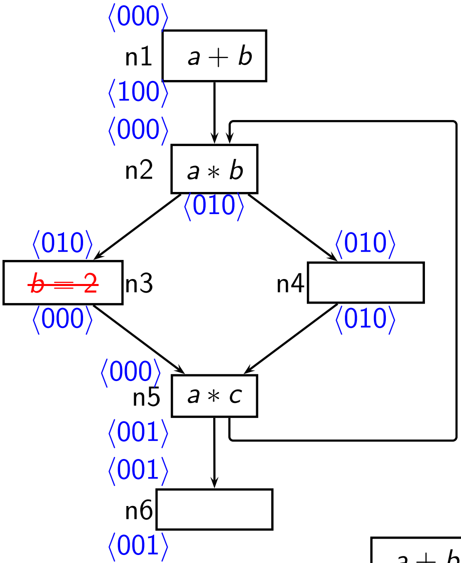


Initial Available Expression Analysis

	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.	0	1	0	0	0	0
2.	0	0	0	1	0	0
3.	0	0	1	0	0	0
4.	0	0	1	1	0	0
5.	0	0	0	0	0	1
6.	0	0	0	0	1	1

$a+b$	$a*b$	$a*c$

Motivating Example

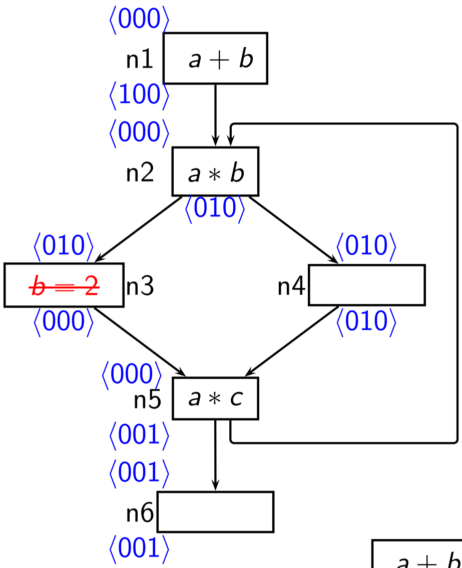


Initial Available Expression Analysis

$\boxed{\text{lower}}$ $\boxed{\text{propagate}}$

$a + b$	$a * b$	$a * c$

Motivating Example

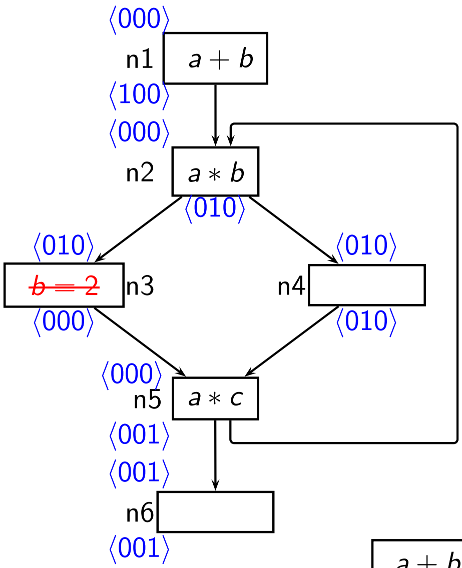


Initial Available Expression Analysis

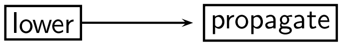
$$\begin{tikzcd}\boxed{\text{lower}} \arrow[r] & \boxed{\text{propagate}}\end{tikzcd}$$

$a + b$	$a * b$	$a * c$

Motivating Example



Initial Available Expression Analysis



Bottom to top change

$a + b$	$a * b$	$a * c$

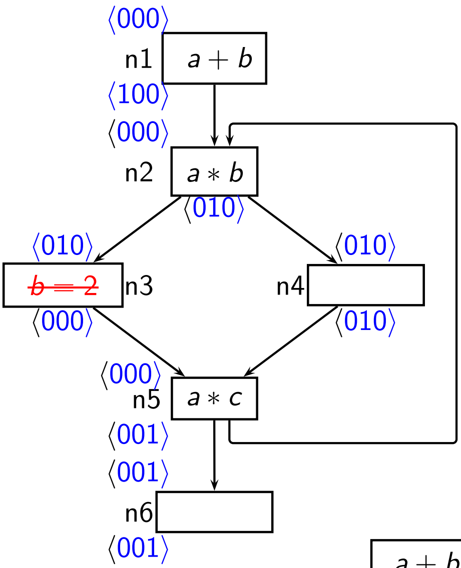
Motivating Example - Step 1

- The data flow values which were 0 and *may* become 1 due to this change

Motivating Example - Step 1

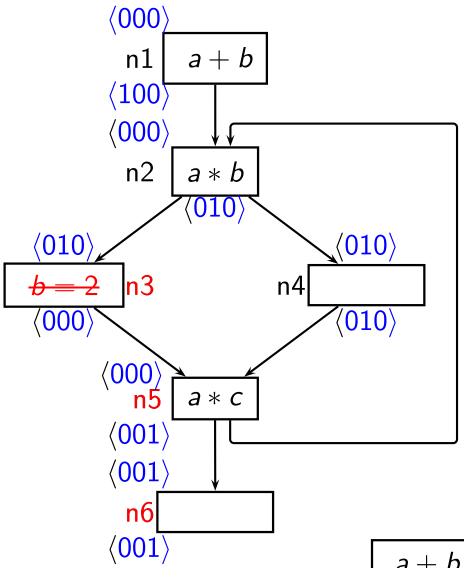
- The data flow values which were 0 and *may* become 1 due to this change
    - Affected region

Motivating Example - Step 1



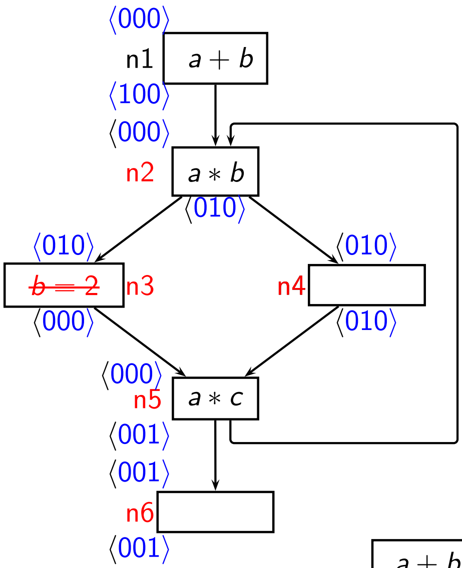
$a + b$	$a * b$	$a * c$

Motivating Example - Step 1



$a + b$	$a * b$	$a * c$

Motivating Example - Step 1

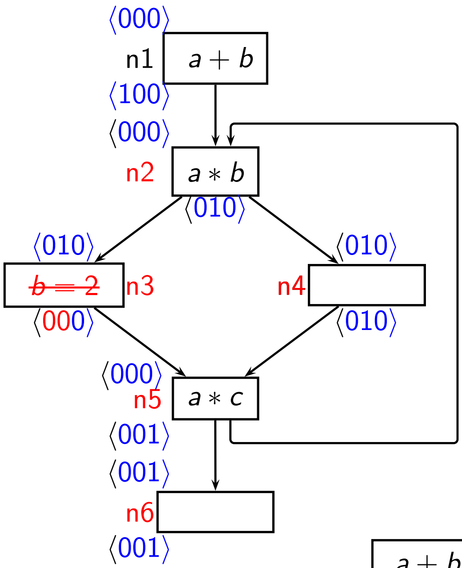


Affected Region

$\langle$ OUT$_3$, IN$_5$, OUT$_5$, IN$_6$, OUT$_6$, IN$_2$, OUT$_2$, IN$_4$, OUT$_4$, IN$_3 \rangle$

$a + b$	$a * b$	$a * c$

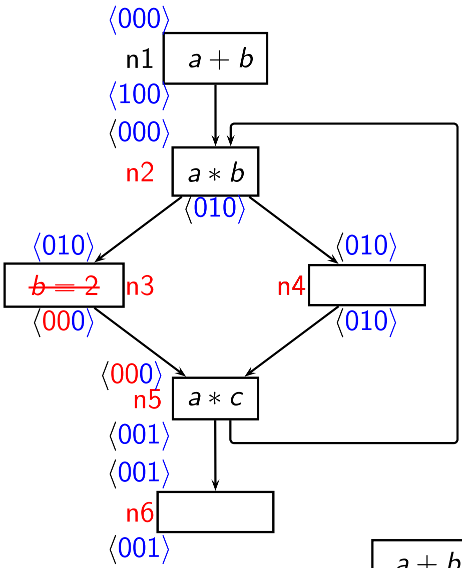
Motivating Example - Step 1



Data flow values which may become 1

$a+b$	$a*b$	$a*c$

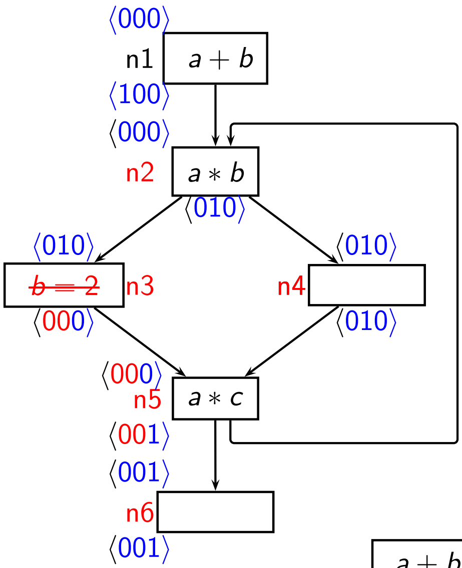
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

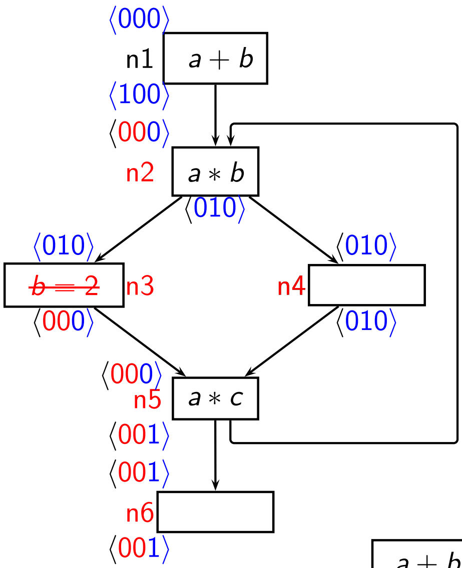
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

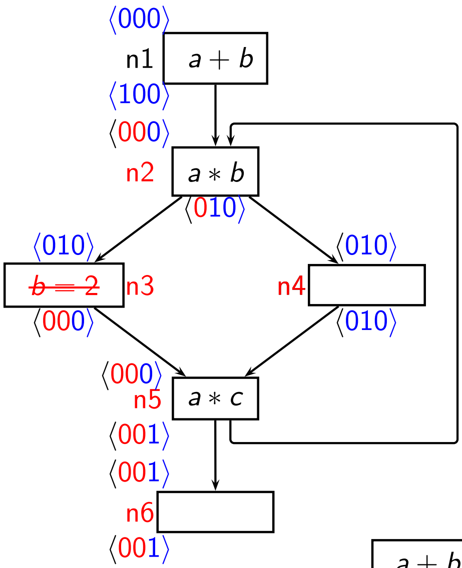
Motivating Example - Step 1



Data flow values which may become 1

$a+b$	$a*b$	$a*c$

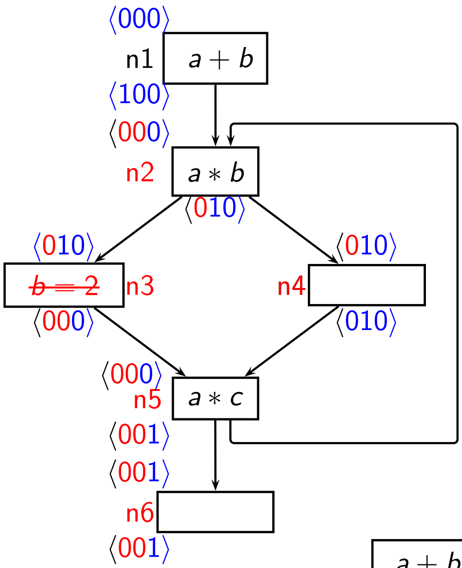
Motivating Example - Step 1



Data flow values which may become 1

$a+b$	$a*b$	$a*c$

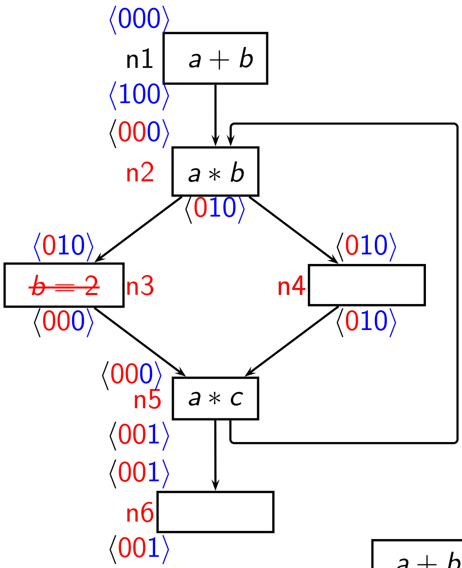
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

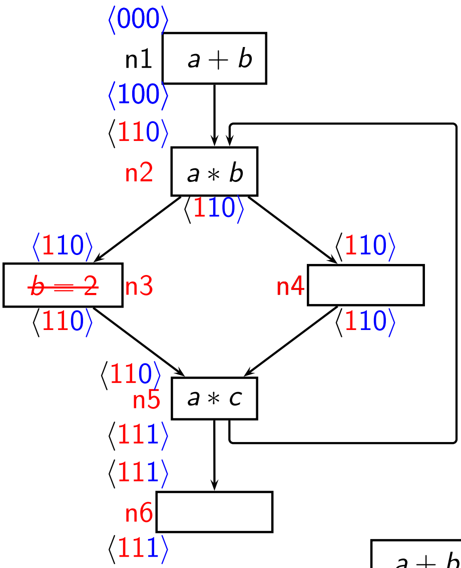
Motivating Example - Step 1



Data flow values which may become 1

$a + b$	$a * b$	$a * c$

Motivating Example - Step 1



Data flow values which may become 1

	$a + b$		$a * b$		$a * c$	
Node	In	Out	In	Out	In	Out
1.						
2.	1	1	1			
3.	1	1		1		
4.	1	1				
5.	1	1	1	1		
6.	1	1	1	1		

$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

- Find out the data flow values which must remain bottom due to the effect of some other property

Motivating Example - Step 2

- Find out the data flow values which must remain bottom due to the effect of some other property
  - Initialize affected region to top.

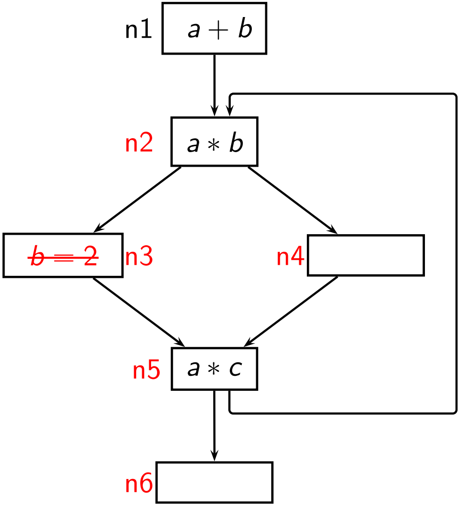
Motivating Example - Step 2

- Find out the data flow values which must remain bottom due to the effect of some other property
  - Initialize affected region to top.
  - Identify boundary nodes

Motivating Example - Step 2

- Find out the data flow values which must remain bottom due to the effect of some other property
  - Initialize affected region to top.
  - Identify boundary nodes
  - Compute values at boundary nodes and propagate them

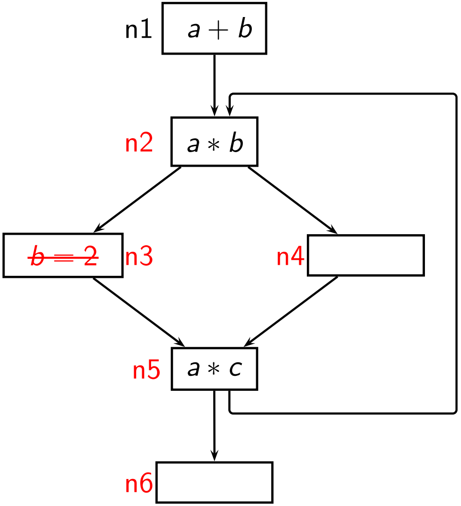
Motivating Example - Step 2



$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

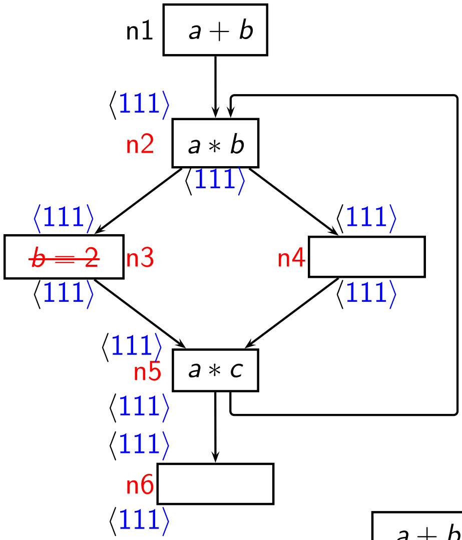
Initialize affected region to top



$a + b$	$a * b$	$a * c$

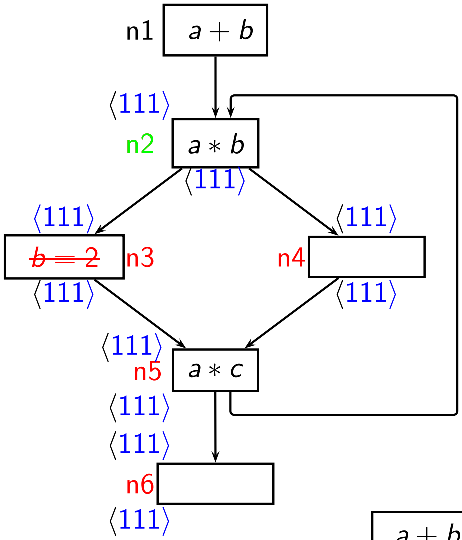
Motivating Example - Step 2

Initialize affected region to top



$a+b$	$a*b$	$a*c$

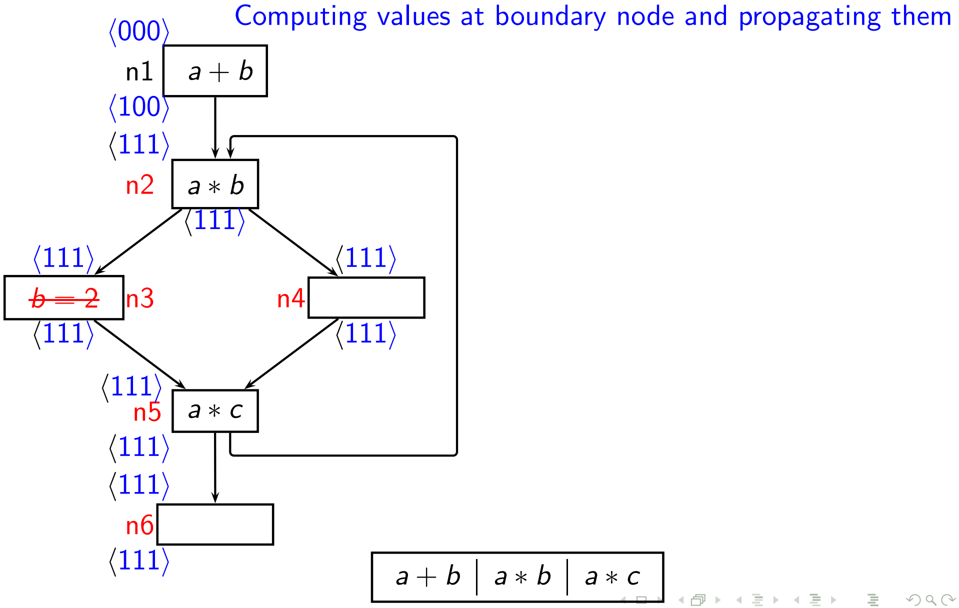
Motivating Example - Step 2



Node 2 is Boundary node

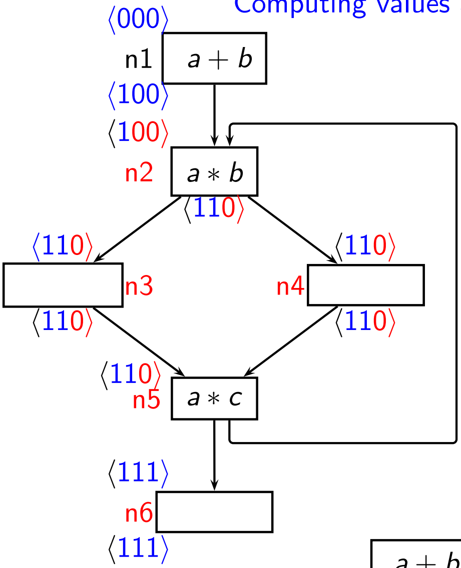
$a+b$	$a*b$	$a*c$

Motivating Example - Step 2



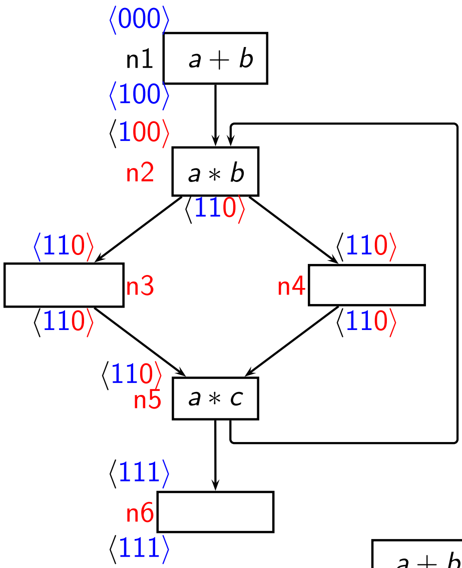
Motivating Example - Step 2

Computing values at boundary node and propagating them



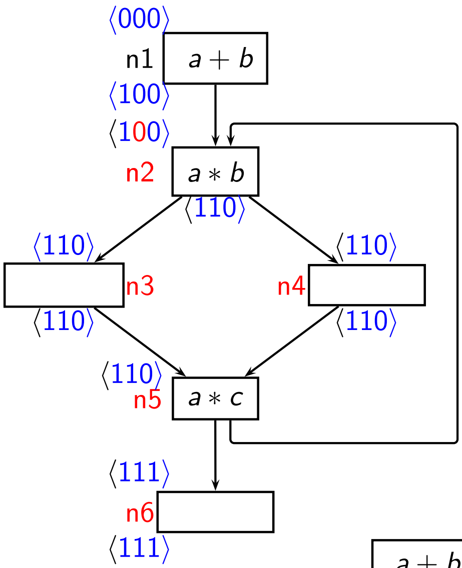
$a+b$	$a*b$	$a*c$

Motivating Example - Step 2



$a + b$	$a * b$	$a * c$

Motivating Example - Step 2

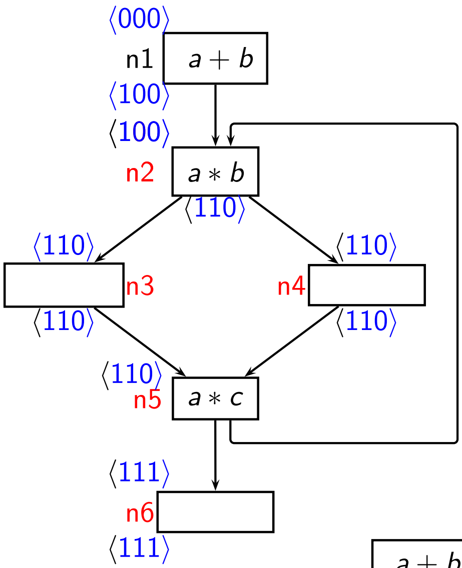


Values which must remain 0

	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.						
2.			0			
3.						
4.						
5.						
6.						

$a+b$	$a*b$	$a*c$

Motivating Example - Step 2



Final values

	$a+b$		$a*b$		$a*c$	
Node	In	Out	In	Out	In	Out
1.	0	1	0	0	0	0
2.	1	1	0	1	0	0
3.	1	1	1	1	0	0
4.	1	1	1	1	0	0
5.	1	1	1	1	0	1
6.	1	1	1	1	1	1

$a + b$	$a * b$	$a * c$

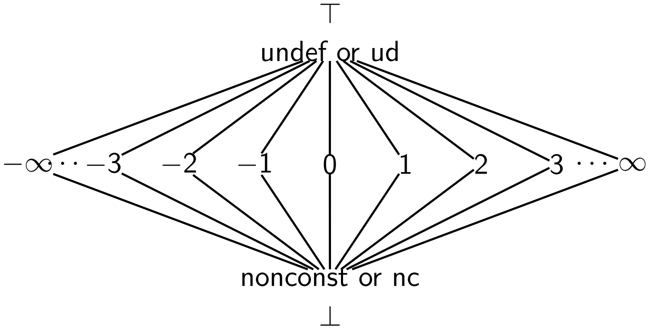
Part III

Incremental Analysis in General Frameworks

Incremental Analysis in General Frameworks

- Consider constant propagation analysis

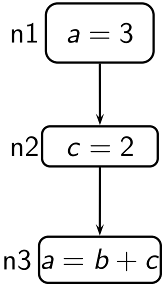
Component lattice for Constant Propagation



Flow functions

- Possible flow functions
    - Top : Similar to raise function
    - Bottom : Similar to lower function
    - Constant : Always produce a constant value
    - Side level : Result depends on the operands of the expression

Constant functions

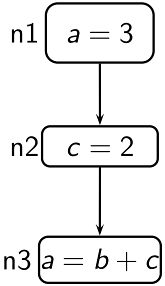


Constant functions

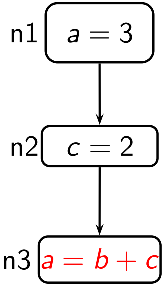
$$\begin{tikzcd}
\text{n1}\ \boxed{a = 3} \arrow[d] \\
\text{n2}\ \boxed{c = 2} \arrow[d] \\
\text{n3}\ \boxed{a = b + c}
\end{tikzcd}$$

Produces Constant values

Side level functions



Side level functions

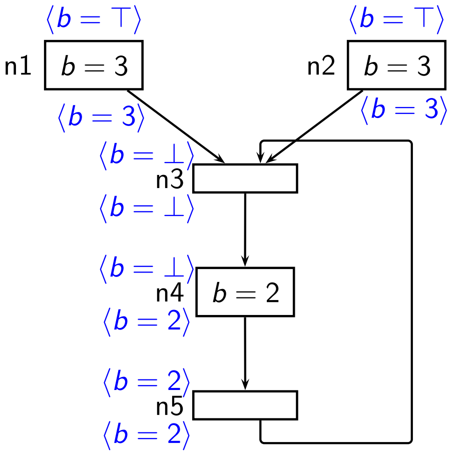


Result depends on the operands

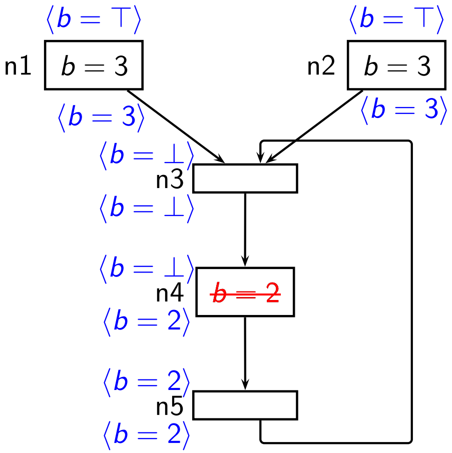
Issues in General Frameworks

- Unlike bit-vector frameworks, when there is a change to bottom:
    - we cannot propagate the change to its neighbouring nodes

Issues in General Frameworks

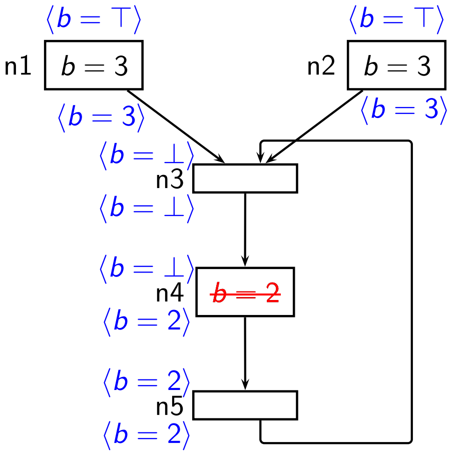


Issues in General Frameworks



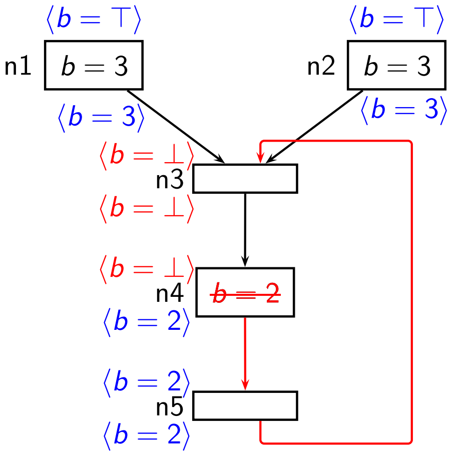
Change to bottom

Issues in General Frameworks



We cannot propagate the change

Issues in General Frameworks



Issues in General Frameworks

- Unlike bit-vector frameworks, we may need to create an affected region even if there is a change to bottom.
- Solution is to create affected region for all kind of changes.

Part IV

Method to Reduce the Size of Affected Region

Method to Reduce the Size of Affected Region

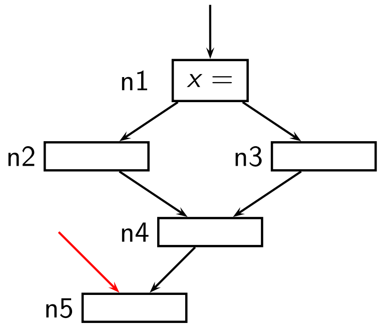
- Based on the observation that some boundary nodes can be characterized by the concept of **Dominance Frontier**.
- Eliminate some boundary nodes from being included in the affected region.

Method to Reduce the Size of Affected Region

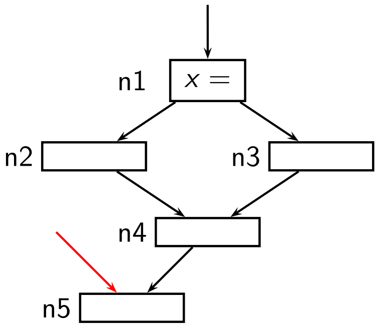
- Let n and m be nodes in CFG. The node n is said to *dominate* m ($n \geq m$), if every path from **Start** to m passes through n.
- If $n \neq m$, then n strictly dominates m, denoted as $n > m$
- **Dominance Frontier**:

$$df(n) = \{m \mid \exists p \in pred(m), (n \geq p \text{ and } n \not> m)\} \tag{1}$$

Method to Reduce the Size of Affected Region

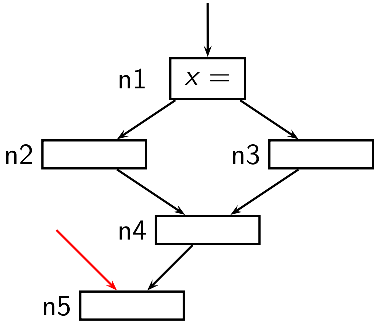


Method to Reduce the Size of Affected Region



$n1$ dominates $n4$

Method to Reduce the Size of Affected Region

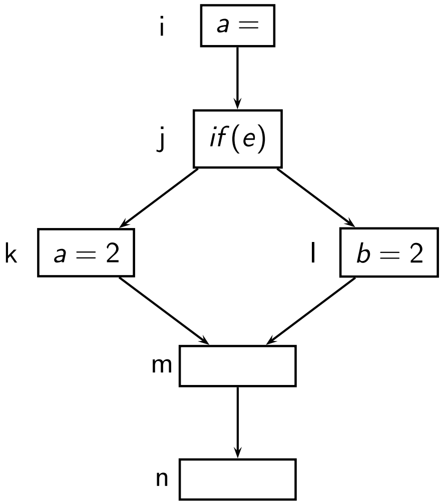


*n*5 is a dominance frontier of *n*1

Method to Reduce the Size of Affected Region

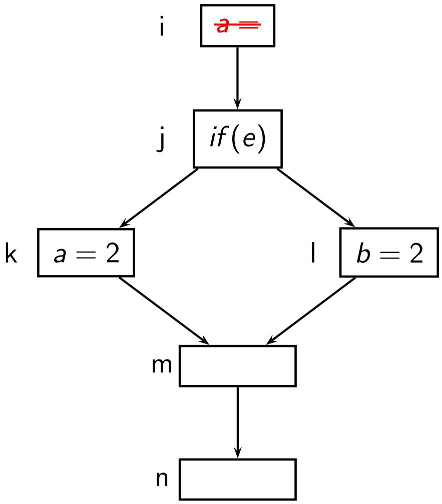
- All Dominance frontier are boundary nodes.
- Vice-versa is not true.

Method to Reduce the Size of Affected Region



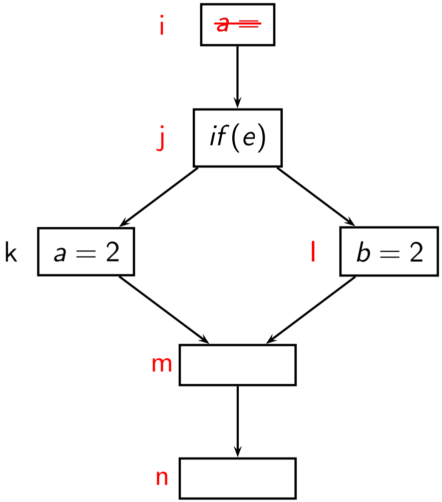
Method to Reduce the Size of Affected Region

Affected region: $< i, j, l, m, n >$



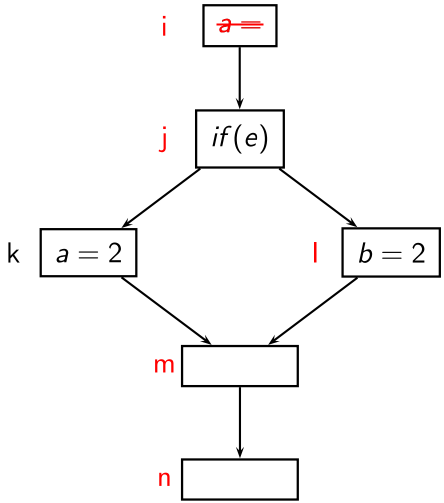
Method to Reduce the Size of Affected Region

Affected region: $< i, j, l, m, n >$



Method to Reduce the Size of Affected Region

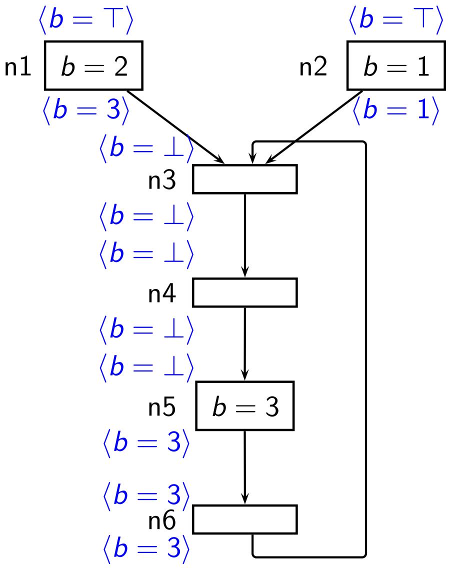
m is a boundary node and is dominated by *i*



Method to Reduce the Size of Affected Region

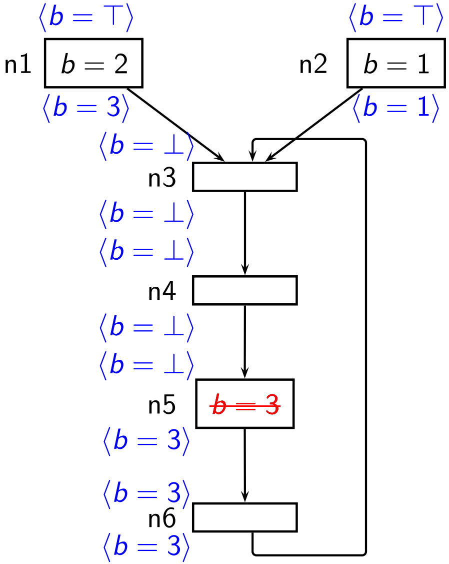
- Possible removal candidates is a dominance frontier of changed node.

Method to Reduce the Size of Affected Region



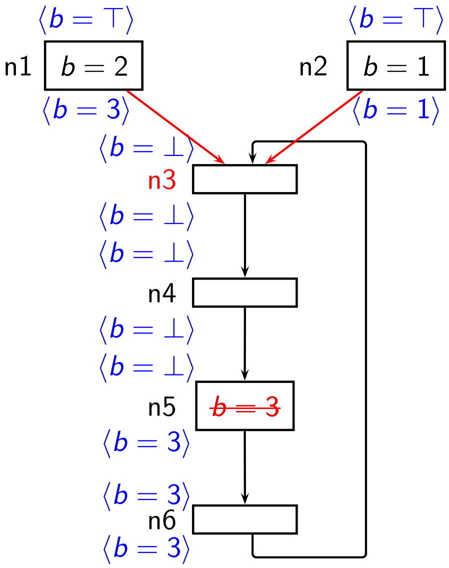
$\langle b = \top \rangle$
n1
$b = 2$
$\langle b = 3 \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \bot \rangle$
n3
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n4
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n5
$b = 3$
$\langle b = 3 \rangle$
$\langle b = 3 \rangle$
n6
$\langle b = 3 \rangle$

Method to Reduce the Size of Affected Region



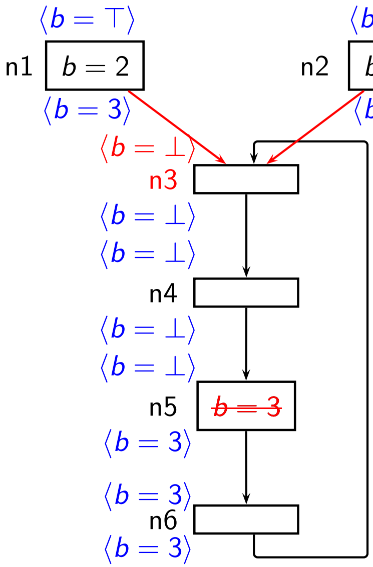
$\langle b = \top \rangle$
$\langle b = \top \rangle$
n1
$b = 2$
n2
$b = 1$
$\langle b = 3 \rangle$
$\langle b = 1 \rangle$
$\langle b = \bot \rangle$
n3
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n4
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n5
~~$b = 3$~~
$\langle b = 3 \rangle$
$\langle b = 3 \rangle$
n6
$\langle b = 3 \rangle$

Method to Reduce the Size of Affected Region



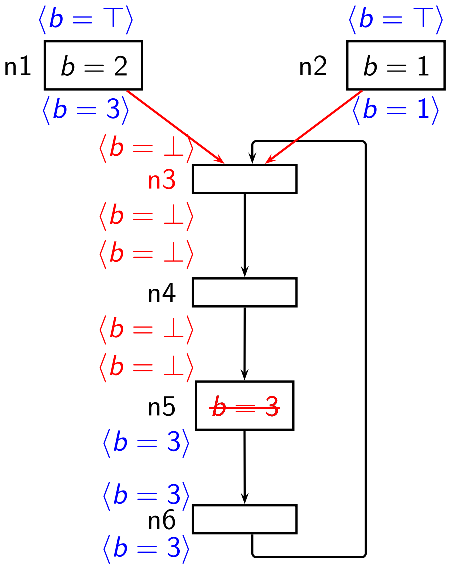
$\langle b = \top \rangle$
n1
$b = 2$
$\langle b = 3 \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \perp \rangle$
n3
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n4
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n5
~~$b = 3$~~
$\langle b = 3 \rangle$
$\langle b = 3 \rangle$
n6
$\langle b = 3 \rangle$

Method to Reduce the Size of Affected Region

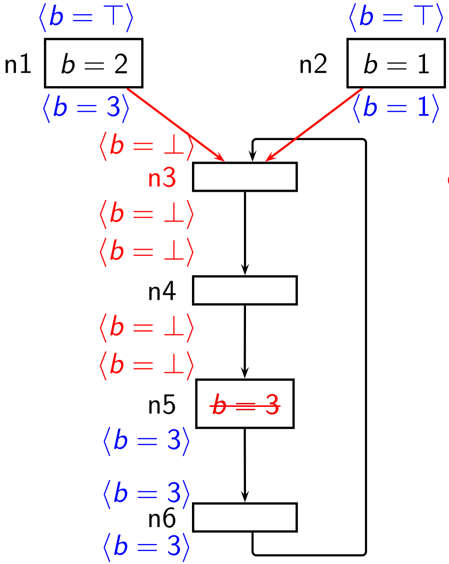


$\langle b = \top \rangle$
$\langle b = \top \rangle$
n1
$b = 2$
n2
$b = 1$
$\langle b = 3 \rangle$
$\langle b = 1 \rangle$
$\langle b = \perp \rangle$
n3
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n4
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n5
~~$b = 3$~~
$\langle b = 3 \rangle$
$\langle b = 3 \rangle$
n6
$\langle b = 3 \rangle$

Method to Reduce the Size of Affected Region


$\langle b = \top \rangle$
n1
$b = 2$
$\langle b = 3 \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \bot \rangle$
n3
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n4
$\langle b = \bot \rangle$
$\langle b = \bot \rangle$
n5
~~$b = 3$~~
$\langle b = 3 \rangle$
$\langle b = 3 \rangle$
n6
$\langle b = 3 \rangle$

Method to Reduce the Size of Affected Region



$\langle b = \top \rangle$
n1
$b = 2$
$\langle b = 3 \rangle$
$\langle b = \top \rangle$
n2
$b = 1$
$\langle b = 1 \rangle$
$\langle b = \perp \rangle$
n3
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n4
$\langle b = \perp \rangle$
$\langle b = \perp \rangle$
n5
~~$b = 3$~~
$\langle b = 3 \rangle$
$\langle b = 3 \rangle$
n6
$\langle b = 3 \rangle$

can eliminate $n3$

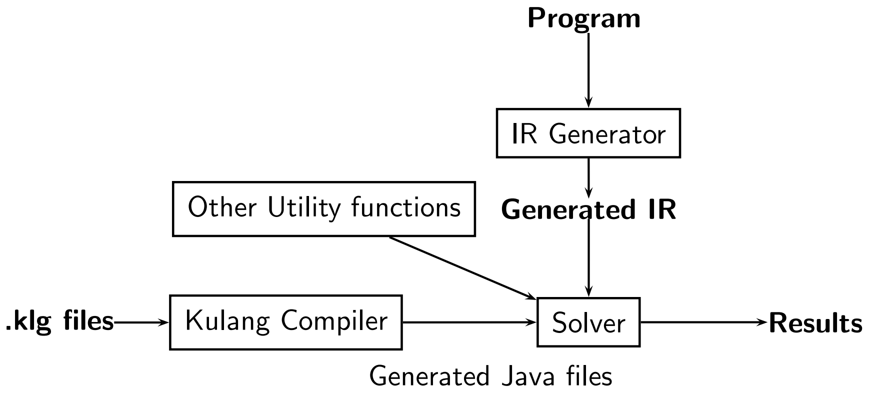
Part V

Overview of PRISM

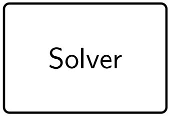
PRISM

- PRISM is a program analyzer generator developed by TATA Research Development and Design Center (TRDDC)

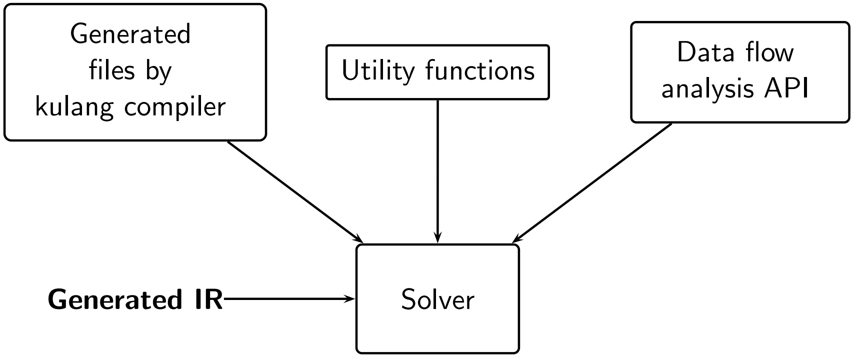
Old Architecture of PRISM



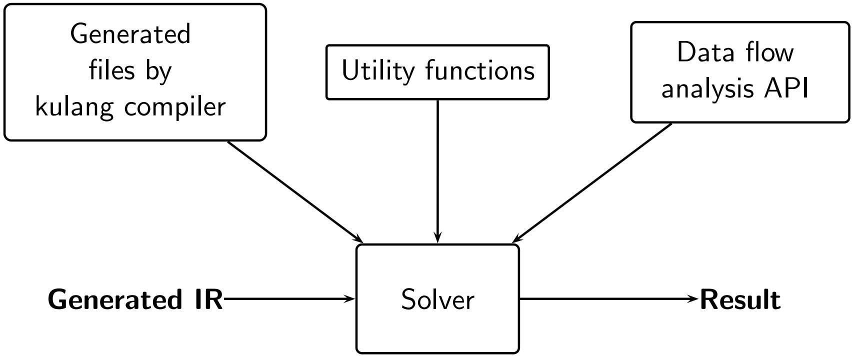
Architecture of Analyzer Generator



Architecture of Analyzer Generator



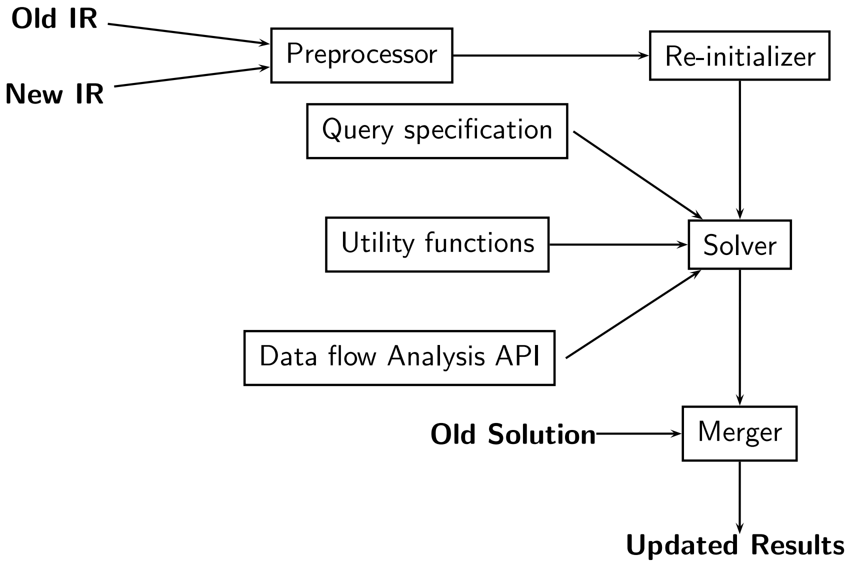
Architecture of Analyzer Generator



Part VI

Incremental Solver

Architecture of Incremental PRISM



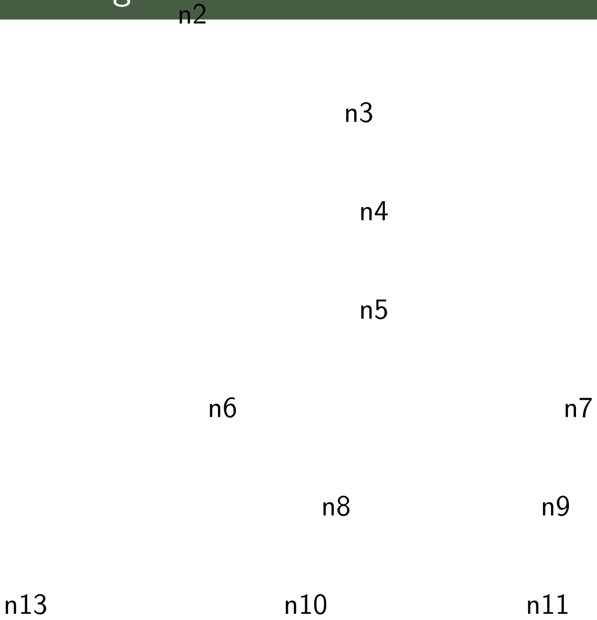
Assumptions

- Pointer information will remain same.
- No change in the context information.
- Declaration of variable haven't change.
- No structural change in the graph.
- A name can refer to a single variable in a program at any given program point.
- past information is stored flow sensitively.

Part VII

Testing

Test Case 1 : *a* is a global variable



Test Case 2 : *a* is a local variable

Test Case 3 : passed as a parameter

Test Case 4 : context sensitive incremental update

Part VIII

Thank You !